スマートフォンアプリケーションの消費電力低減の提案と評価

120430066　坪井 俊也
渡邊研究室

1. はじめに

高齢化社会により高齢者の徘徊行動や孤独死などが問題視されている．我々は，スマートフォンの通信機能とセンサ機能を活用し，見守る側（家族や地域の人など）と見守られる側（高齢者や子どもなど）で，ユーザの位置情報や行動状態などの情報を共有することにより，住民が安心して生活できるシステムとして統合生活支援システム TLIFES（Total LIFE Support system）[1] を提案している．

スマートフォン側アプリケーションは，その消費電力が稼働時間に大きな影響を与える．これまでの省電力化の取り組みにより消費電力は小さくなりつつあるが，歩数計や行動状態の判定は，依然として消費電力が大きく，見守りシステムとして大きな課題となっている．

そこで本稿では，消費電力の大きい加速度センサの CPU 処理に着目し，消費電力低減方法を提案・評価する．

2. 省電力化の経緯と残された課題

TLIFES では，スマートフォンアプリケーションの省電力化のために以下の取り組みがなされてきた [1].

GPS は，屋内のような GPS 衛星の電波が届かない場所においても，必要以上に位置情報を取得しようとして電力を消費する場合がある．そのため，測位開始 10 秒後に衛星数を 4 機以上補足できなかった場合，位置情報が取得できないと判定し，GPS 測位を中断するようにした．

自宅や職場などで移動しない場合は，GPS を起動する必要がない．そのため，加速度値を用いた放置判定，歩数計，乗車判定により,「放置中」「静止中」「乗車中」「歩行中」の 4 種類のユーザの行動状態を判定し，移動中と判定したときにのみ GPS を起動させることとした．さらに,「放置中」には端末をスリープさせ最小限の処理を行うようにした．

以上の取り組みにより，GPS による電力消費は大きく削減された．しかし，行動状態の判定をするために，常に加速度センサを利用するため，CPU の動作に対する消費電力の割合が依然として大きい点が課題となっており，これらの電力を減らす必要がある．

3. 消費電力低減の提案

常に動作している加速度センサの CPU 処理を軽減することにより，消費電力を大きく低減できる．そこで，以下の 2 通りの方法の提案する．

3.1 加速度センサ処理部の C 言語書き換え

現在の TLIFES はすべて Java 言語で記述されているが，一部機能を処理時間が短い C 言語に書き換えることで消費電力低減が図れる．そこで Java 言語空間と C 言語空間とのコード記述の線引きを検討した．

まず，20ms 毎に動作している加速度センサ処理部分を，C 言語に書き換えることとした．次に，Java 言語空間と C 言語空間の遷移をできるだけ無くす必要がある．そこで，加速度センサイベントの取得から行動判定までの一連の処理も含めて，すべて C 言語で記述することとした．

以上より，加速度センサイベントと放置判定，歩数計，乗車判定の各処理をすべて C 言語で実装し直すこととした．

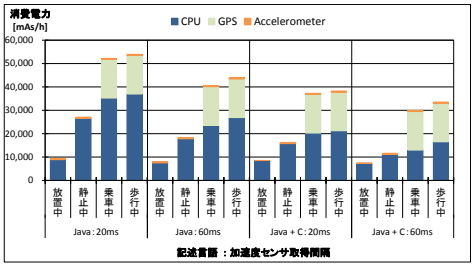


図 1: 既存方式と提案方式の消費電力内訳

3.2 加速度センサ取得間隔の拡大

加速度センサ取得間隔を大きくすることで CPU 計算量を減らし，消費電力低減が図れる．現在の取得間隔は 20ms である．加速度センサ取得間隔を大きくするとナイキスト周波数は小さくなり，ユーザの行動状態の誤判定の原因になる．そこで行動判定に十分な取得間隔を検討した．

歩数計について，人間は 1 秒間に 2 歩程度歩くため，特徴的な周波数は 2Hz 前後である．乗車判定については実測データを観測し，乗車中における加速度値は，電車は 2Hz 前後，車は 1Hz〜8Hz 程度の周波数成分が大きいとわかった．

以上より，ナイキスト周波数は 8Hz 程度でよく，サンプリング間隔は 60ms で十分である．したがって，加速度センサ取得間隔を 60ms まで大きくしても，行動判定結果の精度を維持したまま消費電力が低減できる．

4. 評価

以上の提案を Android 上に実装し，既存の方式と比較評価を行った．Samsung Galaxy Nexus（Android4.2）1 台と，スマートフォンの消費電力を測定できる CORE Power Profiler を用いて，C 言語実装の有無，加速度センサ取得間隔，行動判定の計 16 通りの消費電力を測定した．(図 1)

「静止中」の消費電力は既存（Java : 20ms）と比較して，加速度センサ間隔を 60ms とした場合（Java:60ms）は 68%，センサ処理部を C 言語実装とした場合（Java+C : 20ms）は 60%，2 つの提案を実装した場合（Java+C : 60ms）は 43%となった.「乗車中」「歩行中」の消費電力は，GPS の電力があるため 2 つの提案を実装した場合は既存の約 60%となったが，CPU 処理に着目すると既存の約 40%となった．

以上のことから，提案方式が大幅に消費電力を低減できることが示された．

5. まとめ

本稿では，スマートフォンアプリケーションの消費電力削減のため，一部処理を C 言語に書き換え，センサ取得間隔を大きくする方法を提案した．評価の結果，効果のあることが示された．

参考文献

[1] 加藤大智, 他: TLIFES における省電力化を目的とした位置測位手法の提案と実装, 研究報告コンシューマ・デバイス&システム (CDS), Vol.2013-CDS-6, No.13, pp.1–6, 2013.

スマートフォンアプリケーションの消費電力低減の提案と評価

理工学部　情報工学科　渡邊研究室
120430066　坪井　俊也

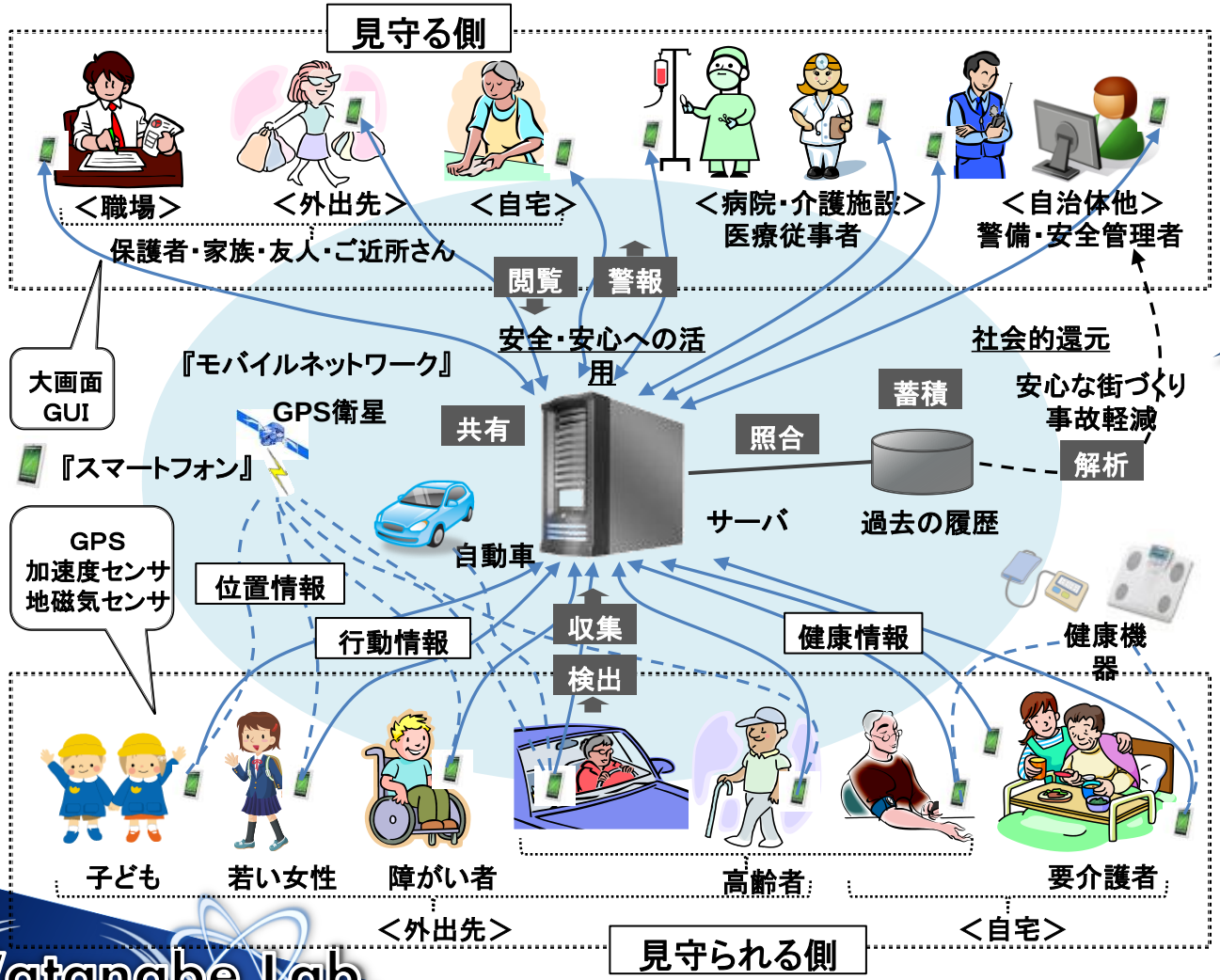
Watanabe Lab.

研究背景

- 高齢化社会の進行
  - 一人暮らしの高齢者の増加
  - 高齢者の徘徊行動
    - 認知症の行方不明者は年間1万人超
- スマートフォンの普及
  - GPSやセンサ，通信など多くの機能を搭載

スマートフォンを用いた見守りシステム
TLIFES(Total LIFE Support system)を提案

TLIFESの概要



徘徊を検出
アラーム
メール送信

- ✓ 位置情報
- ✓ 行動情報

初期からのTLIFESの課題

- スマートフォンアプリケーションの消費電力量
  - 見守りのため定期的にGPSを起動

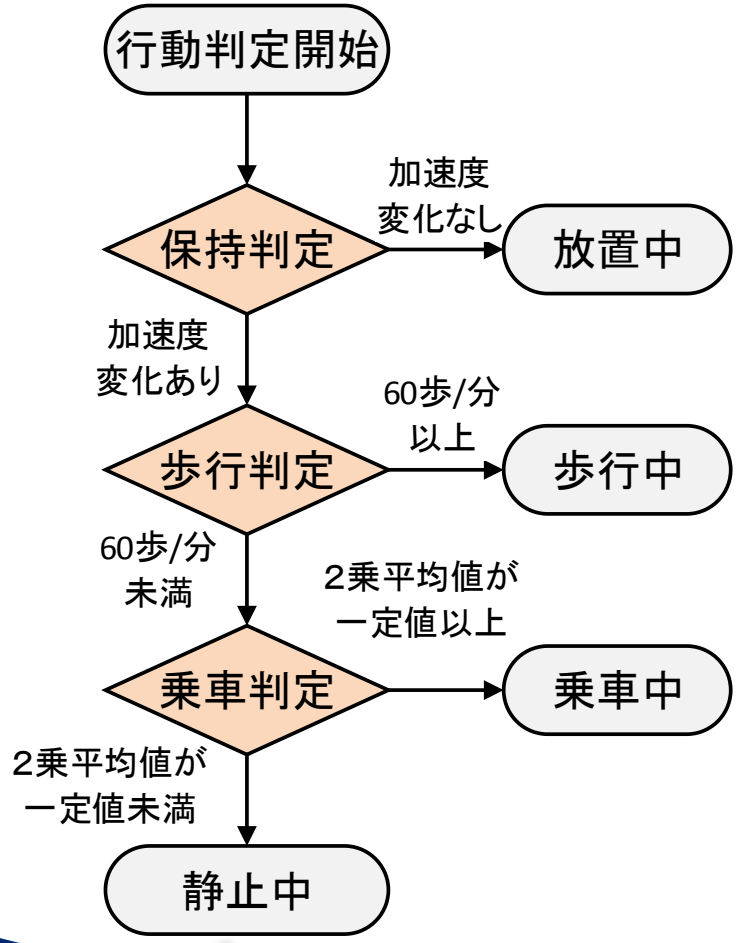
    ⇒スマートフォン稼働時間の低下，バッテリーが1日持たない

- GPS消費電力低減の方策
  1. GPS捕捉衛星数による屋外・屋内判定
     - 電波が届きにくい室内でも測位を行い，電力を消費する

       ⇒屋内であれば即座に測位を終了
  2. 加速度値によるユーザの行動状態判定

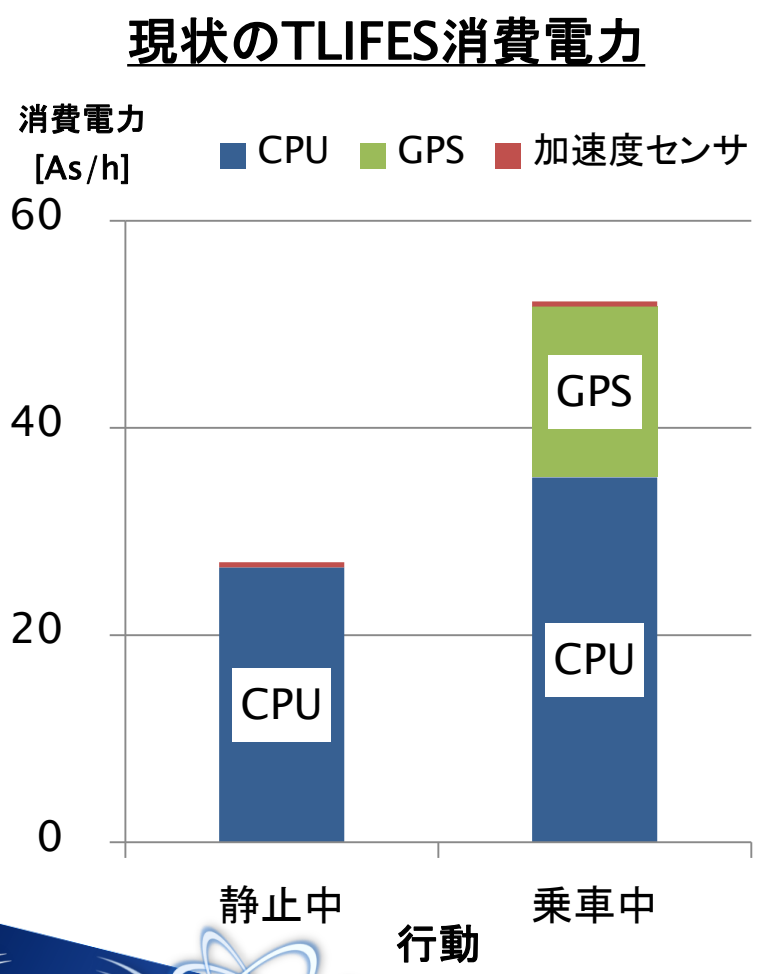
     ⇒歩行や乗車といったユーザの移動中にのみGPSを起動させる

行動判定処理



- 2分ごとに行動判定開始
  - スマートフォンの保持判定
  - 歩数計による歩行判定
  - 加速度値の処理による乗車判定

現状の課題



- GPS電力は低減
- ユーザの行動状態判定
  - CPUが20msごとに
    加速度センサ値を利用
    ⇒CPUの動作にかかる
    消費電力の割合が大

- CPU処理量を重点的に
  低減する2通りの方策を
  提案

提案方式の概要

A) 加速度センサ処理部のC言語書き換え

- Java言語からC言語へ書き換え

B) 加速度センサ取得間隔の拡大

- センサ間隔を20msから60msに拡大

A) センサ処理部のC言語書き換え

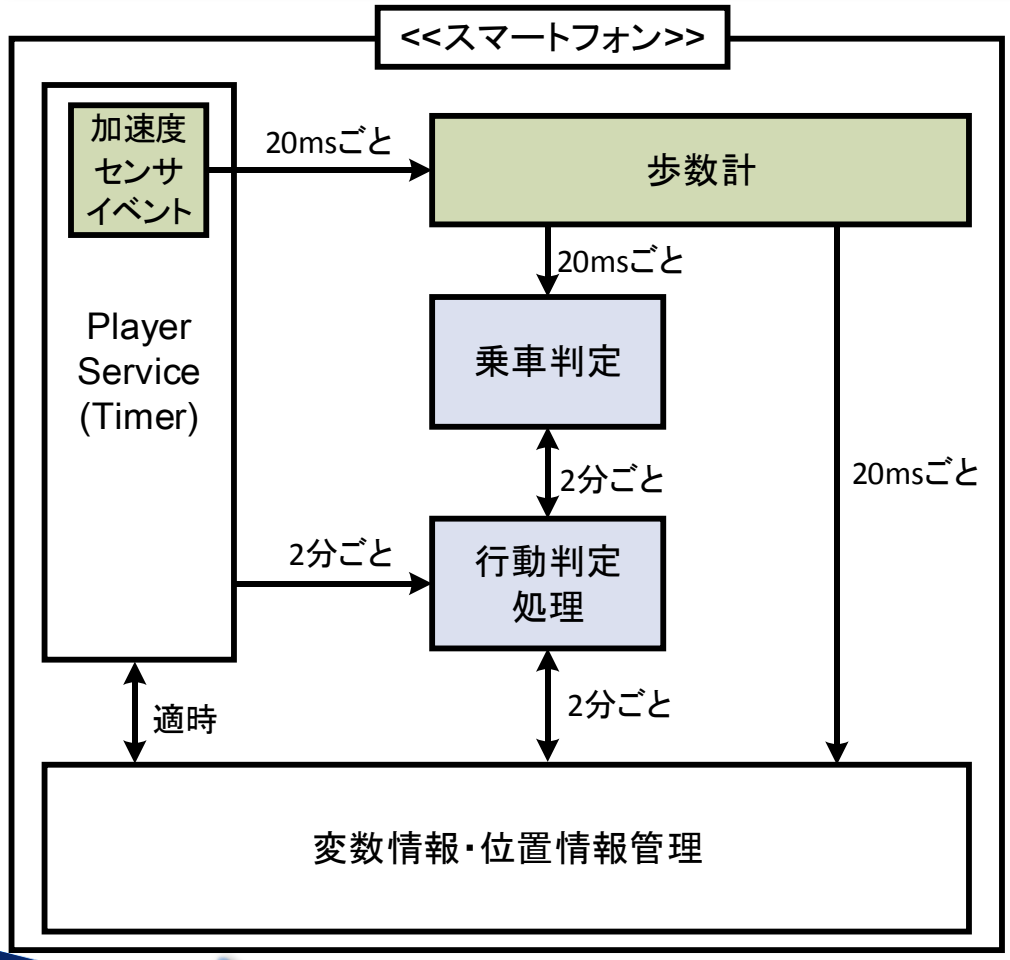
- 現在のTLIFESアプリはAndroid上に実装
  - すべてJava言語で記述

- Java言語実装→C言語実装とした場合，
  実行時間が短く，消費電力が小さい
  ⇒常に動作している処理を書き換えることで大きな効果

- JavaコードとCコード間の遷移を最小限にする必要(※)
  ⇒C言語に書き換える部分を検討

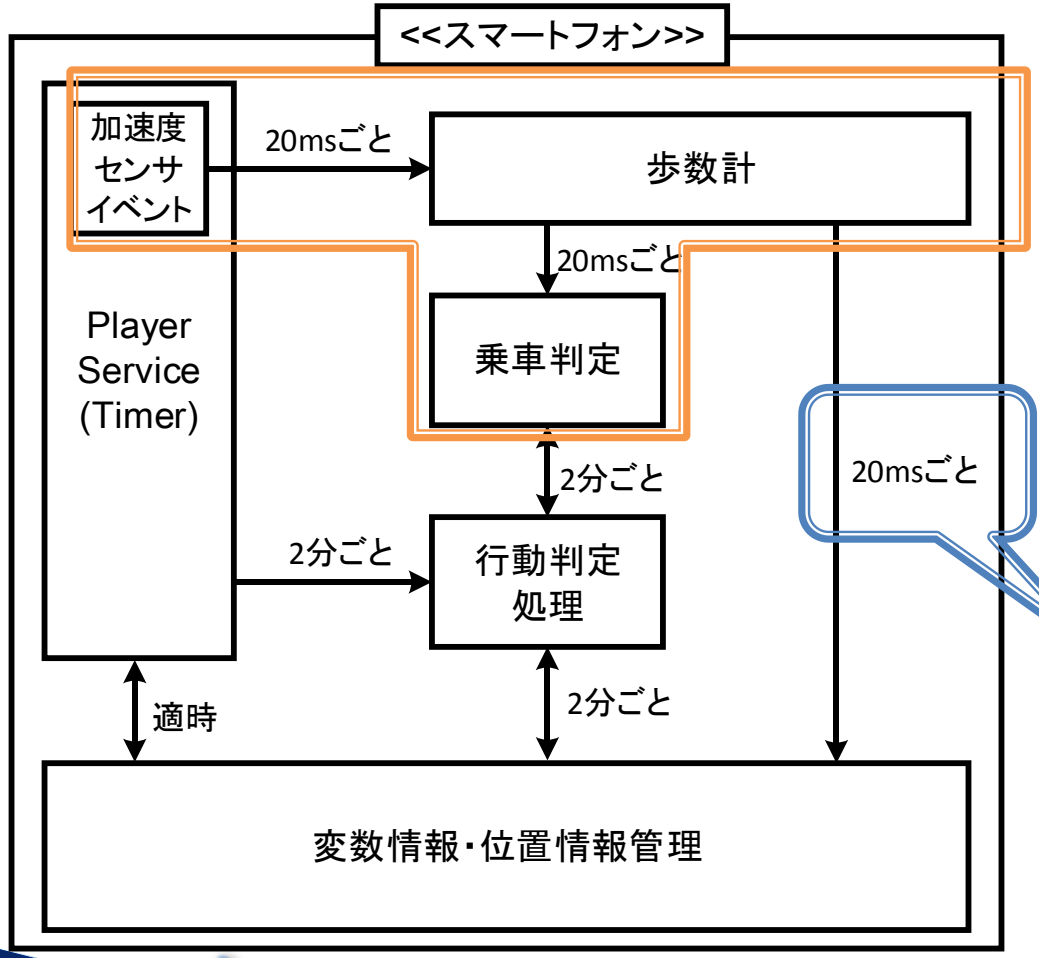
(※)Java Native Interfaceを使用する上でのベスト・プラクティス
http://www.ibm.com/developerworks/jp/java/library/j-jni/index.html

A) センサ処理部のC言語書き換え



- 従来：すべてJava
- 20msごとの処理
  - 加速度センサイベント
  - 歩数計
- 2分ごとの処理
  - 乗車判定
  - 行動判定処理

A) センサ処理部のC言語書き換え

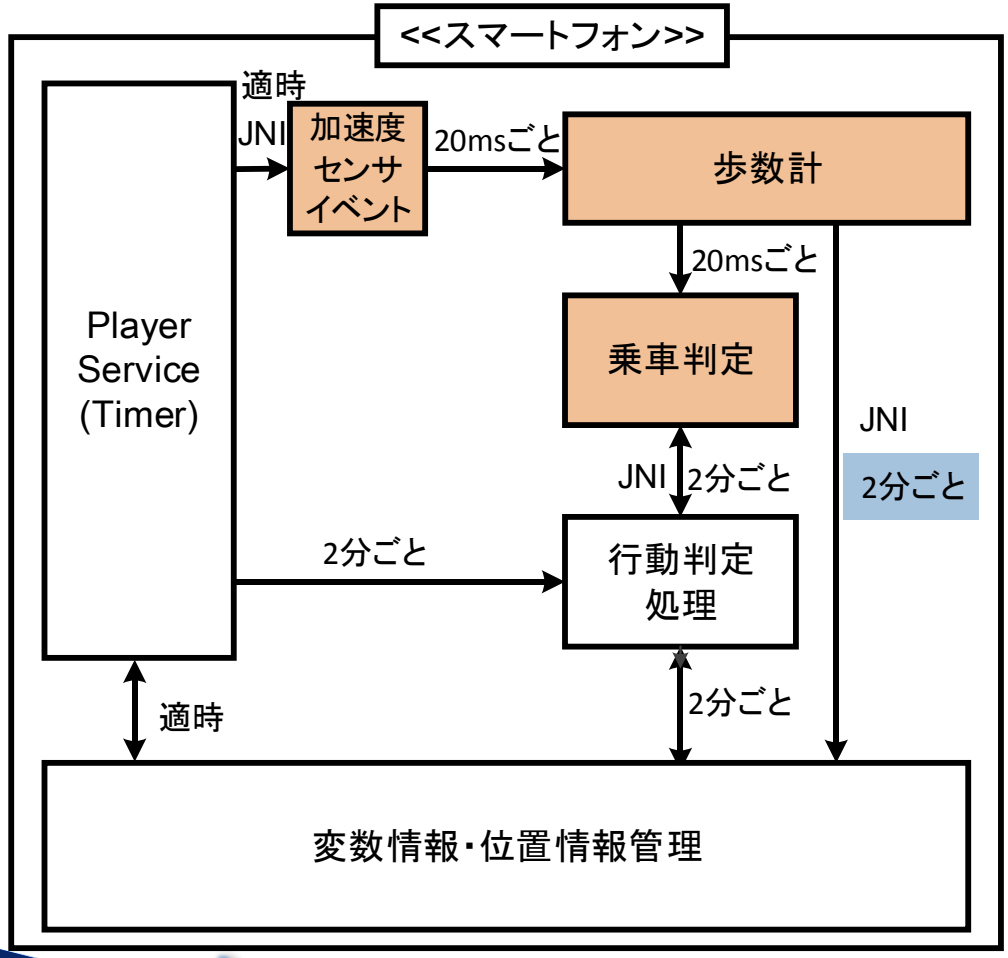


- 20msごとに動作
  - 加速度センサイベント
  - 歩数計
- 20msごとに遷移，計算量のある
  - 乗車判定

⇒C言語に書き換え

- 歩数等の情報
  - サーバ送信間隔 2分ごとに変更

A) センサ処理部のC言語書き換え



- JavaとC間の遷移
  2分ごと
- 書き換え・変更は
  アプリのソース
  コード全体の約1割

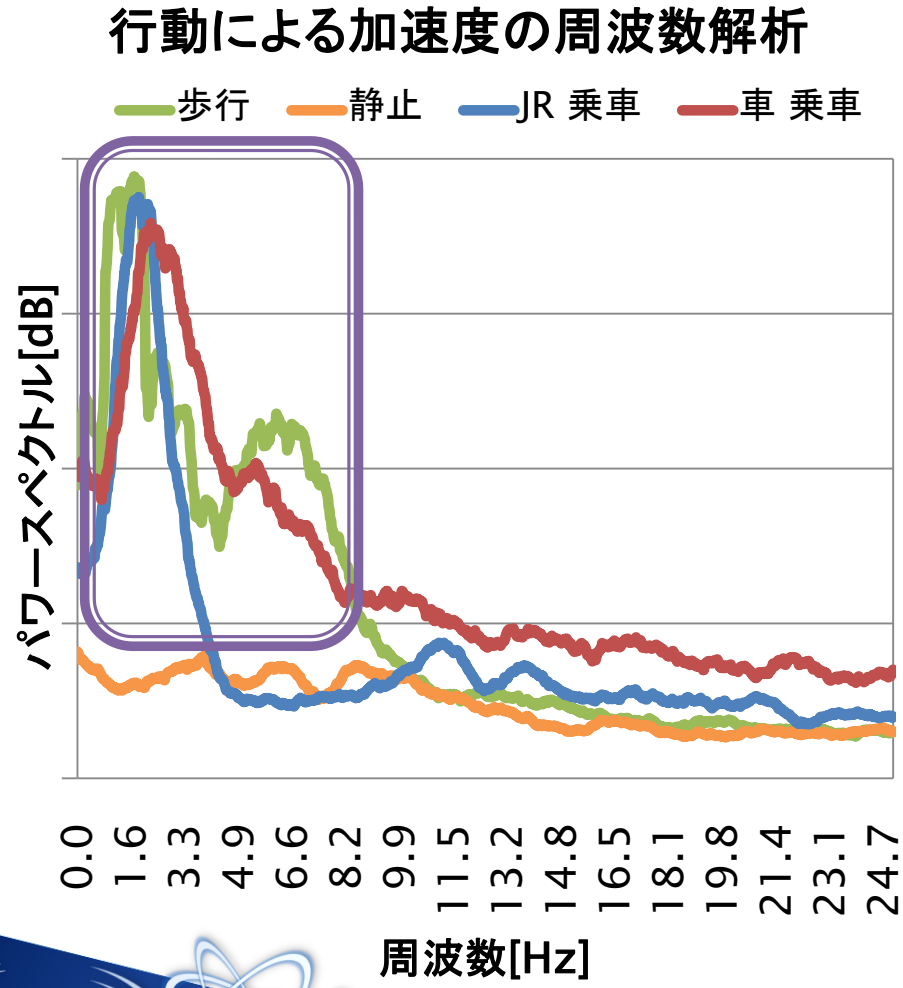
B) 加速度センサ間隔の拡大

- 現状20msの加速度センサ取得間隔

- センサ取得間隔を拡大
  ⇒CPU計算減，消費電力低減
  ⇒ナイキスト周波数が縮小
  - ユーザの行動状態の誤判定の原因

- 行動判定に十分な取得間隔を検討
  - 行動による加速度値の周波数成分の解析

B）加速度センサ間隔の拡大



- 1Hz～8Hz前後
  特徴的な周波数
  - ナイキスト周波数8Hz
  - サンプリング間隔
    $= \frac{1}{8[Hz]\times 2} = 62.5[\mathrm{ms}]$

- 行動判定精度を保ち
  センサ間隔を60ms
  に拡大

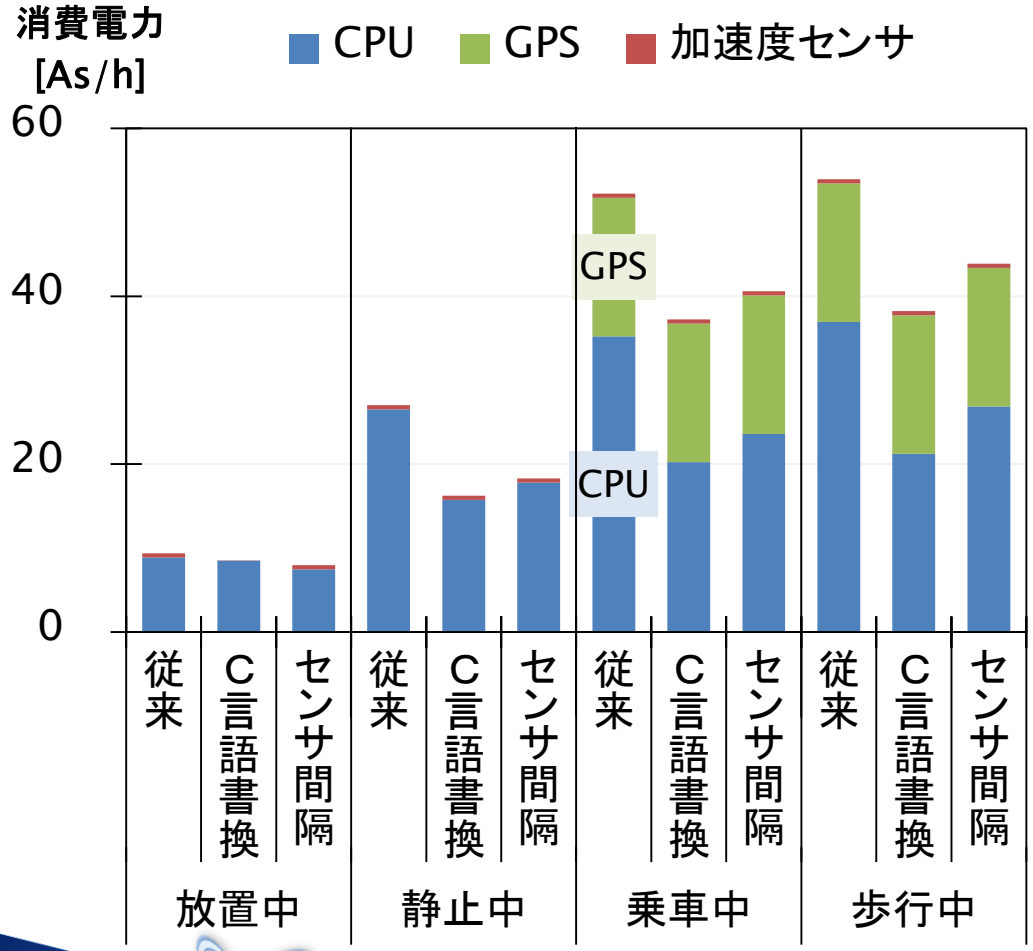
評価

- 評価方法
  - 提案方式をAndroid上にコーディング・実装して，アプリケーションの消費電力量を測定
- 測定方法
  - 使用端末：Samsung Galaxy Nexus（Android4.2）1台
  - 測定アプリケーション：CORE Power Profiler (※)
- 測定項目
  - 従来方式
  - C言語書換のみ
  - センサ間隔拡大のみ
  - 両方実装した提案方式

  - 放置中
  - 静止中
  - 乗車中
  - 歩行中
  - 1項目ごとに3時間測定を行い，1時間単位の平均値を用いた

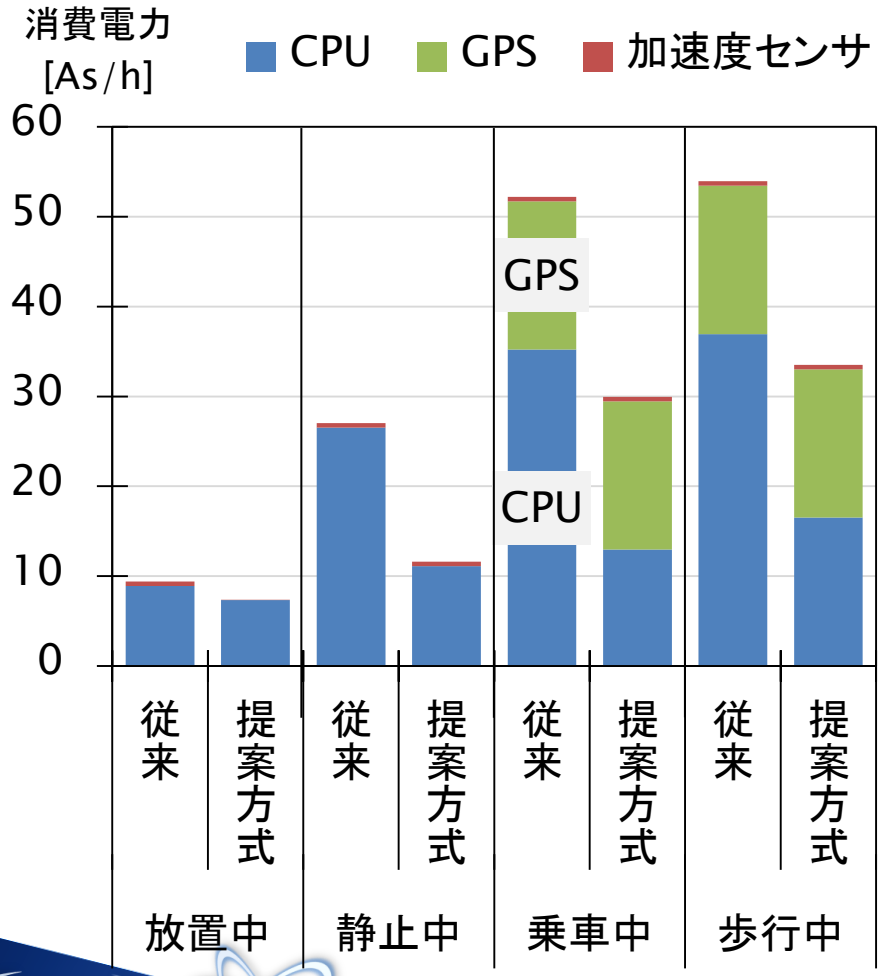
(※)消費電力測定アプリ：
http://www.core.co.jp/product/smartdevice/outline/corepowerprofiler.html

評価—各提案の消費電力測定結果



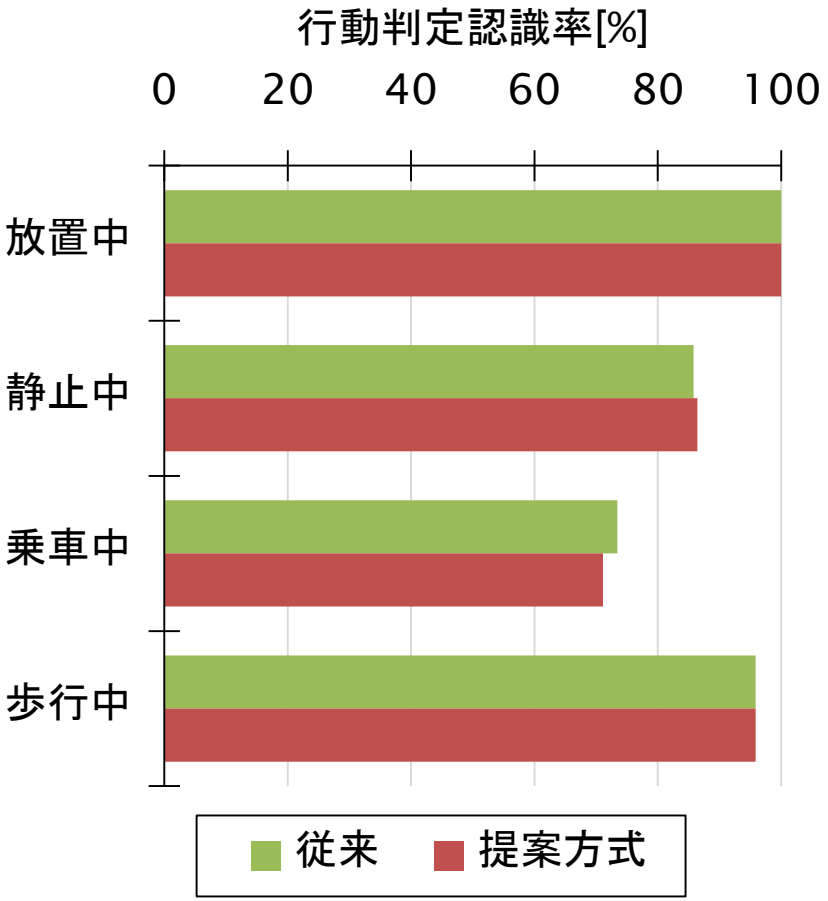
- 従来と各提案を比較
- 静止中
  - C言語書換：約60%
  - センサ間隔拡大：約67%
- 乗車中，歩行中
  - C言語書換：約70%
  - センサ間隔拡大：約80%

評価―提案方式の消費電力測定結果



- 従来と提案方式の比較
- 静止中
  - 従来の約43%
- 乗車中・歩行中
  - GPSの電力が加算
    全体では約60%
  - CPUに着目
    従来の約37%，約45%
- CPUの電力
  - 従来比**約40%**

評価―行動判定認識率比較



- 従来と提案方式を比較
  - 提案方式は従来と同等の認識率と確認
- 行動判定認識率を維持，消費電力低減

まとめ

- 現状の課題
  - CPUの動作にかかる消費電力の割合が大
- 消費電力低減の方策の提案
  - A) 加速度センサ処理部のC言語書き換え
  - B) 加速度センサ取得間隔の拡大
- 評価
  - Android上に実装を行い，行動判定精度を維持しながら，CPU動作の消費電力は従来の約40%となることを確認

補足資料

- 以下，補足資料

TLIFESの概要

- スマートフォンの通信機能とセンサ機能を活用し、ユーザが情報を共有できるシステム
- ユーザ全員がスマートフォンを所持
- ユーザの行動情報、位置情報、歩数などを収集
- 情報は定期的にサーバへ送信しデータを蓄積
- 許可されたメンバはデータの閲覧可能
- 過去の履歴と比較し，ユーザの異常があると判断したらアラームを送信

今後の課題

- 提案方式では
  - CPUがスリープせず常に起動している
  - 基礎消費電力は依然大きい
- Hardware Sensor Batchingを利用
  - CPUをスリープさせたままハードウェアがセンサ値を取得可能

提案にかかる変更部分の各ステップ数

- 総ステップ数
  - プログラムのソースコードの総行数
- 実ステップ
  - 総ステップのうち空行やコメント行を除いた行数

	総ステップ数	実ステップ数
従来方式	10,589	6,371
・内書き換え/修正	877	597
提案方式	11,278	6,719
・内修正/追加	1,367	821